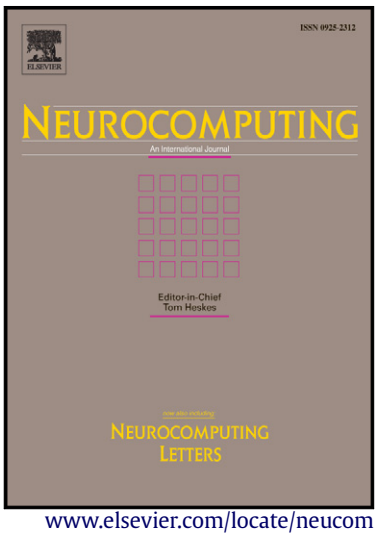
A multi-output two-stage locally regularized model construction method using the extreme learning machine

Du, D., Li, K., Li, X., Fei, M., & Wang, H. (2014). A multi-output two-stage locally regularized model construction method using the extreme learning machine. *Neurocomputing*, *128*, 104-112. https://doi.org/10.1016/j.neucom.2013.03.056

Published in:
Neurocomputing

Document Version:
Peer reviewed version



www.elsevier.com/locate/neucom

A multi-output two-stage locally regularized model construction method using the extreme learning machine

Dajun Du, Kang Li, Xue Li, Minrui Fei, Haikuan Wang

PII: S0925-2312(13)01006-0
DOI: http://dx.doi.org/10.1016/j.neucom.2013.03.056
Reference: NEUCOM13730

To appear in: *Neurocomputing*

Received date: 5 September 2012
Revised date: 25 February 2013
Accepted date: 4 March 2013

Cite this article as: Dajun Du, Kang Li, Xue Li, Minrui Fei, Haikuan Wang, A multi-output two-stage locally regularized model construction method using the extreme learning machine, *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2013.03.056

A multi-output two-stage locally regularized model construction method using the extreme learning machine

Dajun Du[a,b], Kang Li[b,*], Xue Li[a], Minrui Fei[b], Haikuan Wang[b]

[a] *Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronical Engineering and Automation, Shanghai University, Shanghai 200072, China*

[b] *School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT9 5 AH, UK.*

Abstract

This paper investigates the construction of linear-in-the-parameters (LITP) models for multi-output regression problems. Most existing stepwise forward algorithms choose the regressor terms one by one, each time maximizing the model error reduction ratio. The drawback is that such procedures cannot guarantee a sparse model, especially under highly noisy learning conditions. The main objective of this paper is to improve the sparsity and generalization capability of a model for multi-output regression problems, while reducing the computational complexity. This is achieved by proposing a novel multi-output two-stage locally regularized model construction (MTLRMC) method using the extreme learning machine (ELM). In this new algorithm, the non-linear parameters in each term, such as the width of the Gaussian function and the power of a polynomial term, are firstly determined by the ELM. An initial multi-output LITP model is then generated according to the termination criteria in the first stage. The significance of each selected regressor is checked and the insignificant ones are replaced at the second stage. The proposed method can produce an optimized compact model by using the regularized parameters. Further, to reduce the computational complexity, a proper regression context is used to allow fast implementation of the proposed method. Simulation results confirm the effectiveness of the proposed

*Corresponding author, *E-mail address*: k.li@qub.ac.uk

technique.

Keywords: Extreme learning machine; multi-output linear-in-the-parameters (LITP) model; regularization; two-stage stepwise selection.

1. Introduction

The modeling and identification of multi-input multi-output (MIMO) dynamic systems have been used in many industrial applications [1]-[2]. A conventional approach is to identify a multi-input single-output model for each output separately and then combine every individual model to produce a final MIMO model [2]. However, if there are common or correlated parameters for different output variables, then performing identification on all outputs simultaneously may lead to better and more robust models.

Some research has been reported on the simultaneous identification of the MIMO systems. For example, multi-innovation stochastic gradient [3] and hierarchical least squares algorithms [4] have been proposed for multi-output systems. Gradient-based and least-squares-based iterative estimation algorithms for MIMO systems have also been proposed [5]. Integrating support vector regression and annealing dynamical learning algorithm, a robust approach was developed to optimize a radial basis function (RBF) network for the identification of MIMO systems [1].

A popular alternative approach is to formulate the modeling of MIMO systems as a linear-in-the-parameters (LITP) problem (e.g. support vector machine (SVM) model [1] or RBF neural model [6]), for which some well-known solutions can be applied. For a LITP model, its performance critically depends upon the determination of the nonlinear parameters in each model term, such as the width of a Gaussian function or the fractional power of a polynomial term. A conventional strategy is to randomly select some input data points as the RBF centers [7], which may unfortunately produce a network with poor performance. To tackle this, clustering techniques have been introduced for the center location [8]. In contrast to such traditional computational intelligence techniques, the extreme learning machines (ELM) has been proposed in [9]-[11]. It applies random computational nodes in the hidden layer that do not need to be tuned. The hidden layer thus has the fixed parameters, allowing the output weights to be solved using the least-squares.

There are some well-known methods for the identification of LITP models. These include the popular forward orthogonal least squares (OLS) [12] and the fast forward recursive algorithm (FRA), which are used to select candidate terms (regressors) based on their contributions to maximizing the model error reduction ratio, and for RBF neural networks, all the training samples are usually used in generating the candidate terms. The OLS algorithm has also been extended for selecting the centers for multi-output RBF neural networks [13]. Further, recursive OLS algorithm is also employed to select the centers for multi-output RBF neural networks [6]. Unlike OLS that uses QR decomposition on the regression matrix, the recently FRA [14], [15] proposes a regression context based on which fast selection of the model structure and fast estimation of model parameters are achievable. It has been shown that FRA requires much less computational effort and is also numerically more stable than some of the alternatives. The FRA method has been further extended to construct multi-output RBF neural model [16].

The above forward selection methods only provide an efficient pathway for the identification of MIMO systems. However, these methods do not consider how to control the model complexity. In general, a model with too many parameters will tend to overfit the training set and therefore fail to generalize to the test set. Conversely, a model with too few parameters will underfit the training data and hence achieve poor predictive power on both the training data and the test set. Ideally, a sparse learner balances model complexity against training set size, with the goal of balancing between under- and over-fitting. The benefits of the resulting sparse model include improved generalization capability and robustness to new test data and greater efficiency [17], [18]. The regularization approach [19]-[21] is a useful technique to enforce the sparsity of MIMO model and to overcome the over-fitting problem. However, the regularization parameters have to be tuned to obtain satisfactory performance [22]. According to the Bayesian learning theory [19], [20], a regularization parameter is equivalent to the ratio of the related hyperparameter to a noise parameter. Compared with traditional regularization methods, the Bayesian approach provides a rigorous framework for automatic adjustment of the regularization parameters to their near-optimal values. This is achieved by marginalizing the hyperparameters when making inferences, and no validation data set is needed. The Bayesian evidence procedure has also been incorporated into multi-output OLS (MOLS) [24].

In this paper, the extreme learning machine and regularized technique

are introduced into the recently proposed two-stage stepwise selection algorithm [15], leading to a novel multi-output two-stage locally regularized model construction (MTLRMC) method. In this new algorithm, the nonlinear parameters in each term, such as the width of the Gaussian function and the power of a polynomial term, are firstly determined by the ELM. An initial multi-output LITP model is generated according to the termination criteria in the first stage. The significance of each selected regressor is then checked and the insignificant ones are replaced at the second stage. The proposed method can produce an optimized compact model by the regularization parameters. Further, to reduce the computational complexity, a proper regression context is defined which allows fast implementation of the proposed method.

The paper is organized as follows. Section 2 gives some preliminaries on multi-output Linear-in-the-parameters model, determining the centres and widths using the ELM and the parameter estimation of multi-output Linear-in-the-parameters models. Section 3 presents the proposed multi-output two-stage locally regularized model construction method, including the net error reduction to the regularized cost function, stage 1-forward model selection, stage 2-backward model refinement, complete algorithm, and computational complexity analysis. Simulation results are presented in Section 4, followed by the concluding remarks in Section 5.

2. Preliminaries

2.1. Multi-output Linear-in-the-parameters model

Consider a discrete-time multivariable nonlinear system with m inputs and g outputs

$$\begin{aligned} y_i(t) &= f(y_1(t-1),\ldots,y_1(t-n_y^1),\cdots,y_g(t-1),\ldots,y_g(t-n_y^g),\ldots, \\ &\quad u_1(t-1),\ldots,u_1(t-n_u^1),\ldots,u_m(t-1),\ldots,u_m(t-n_u^m)), \\ &= f(x(t)), \end{aligned} \tag{1}$$

where $y_i(t), i = 1,\ldots,g$ and $u_j(t), j = 1,,\ldots,m$ are the system output and input; n_y^i and n_u^j are the corresponding maximal lags of the i^{th} output and j^{th} input; $x(t) = [y_1(t-1),\ldots,u_m(t-n_u^m)]^T$ is model "input" vector; $f(\cdot)$ is some unknown nonlinear function, g and m are the number of system outputs and inputs, respectively.

Suppose a multi-output linear-in-the-parameters (LITP) model is used to represent (1) such that

$$y_i(t) = \hat{y}_i(t) + \varepsilon_i(t) = \sum_{j=1}^{M} \theta_{j,i}\varphi_j(x(t)) + \varepsilon_i(t), i = 1, \ldots, g, \tag{2}$$

where $\theta_{j,i}$ are the model weights, $\varepsilon_i(t)$ is the error between $y_i(t)$ and the i^{th} model output $\hat{y}_i(t)$ and $\{\varepsilon_i(t)\}$ is assumed to be a white sequence, M is the number of basis functions, $\varphi_j(\cdot)$ is a known nonlinear basis function, such as Gaussian, polynomial or B-spline functions, and so on. If a Gaussian kernel function $\phi(x, c_j, \sigma_j) = \exp(-\left\|x - c_j\right\| / \sigma_j^2)$ is used, then (2) can be re-written as

$$y_i(t) = \sum_{j=1}^{M} \theta_{j,i}\varphi_j(\left\|x(t) - c_j\right\|; \sigma_j) + \varepsilon_i(t), i = 1, \ldots, g. \tag{3}$$

Suppose N data samples $\{x(t), Y(t)\}_{t=1}^{N}$ are used for model identification, (3) can be re-written in matrix form as

$$Y = \Phi\Theta + \Xi, \tag{4}$$

where $Y = [y_1, y_2, \ldots, y_g] \in \Re^{N\times g}$ with column vectors $y_i = [y_i(1), y_i(2), \ldots, y_i(N)]^T$, $i = 1, 2, \ldots, g$; $\Phi = [\phi_1, \phi_2, \ldots, \phi_M] \in \Re^{N\times M}$ with column vectors $\phi_i = [\phi_i(\left\|x(1) - c_i\right\|, \sigma_i), \ldots, \phi_i(\left\|x(N) - c_i\right\|, \sigma_i)]^T$, $i = 1, 2, \ldots, M$; $\Theta = [\theta_1, \ldots, \theta_g] \in \Re^{M\times g}$ with column vectors $\theta_i = [\theta_{1,i}, \ldots, \theta_{M,i}]^T$, $i = 1, \ldots, g$; $\Xi = [\varepsilon_1, \ldots, \varepsilon_g] \in \Re^{N\times g}$ with column vectors $\varepsilon_i = [\varepsilon_i(1), \ldots, \varepsilon_i(N)]^T$, $i = 1, \ldots, g$.

2.2. Determining the centers and widths using the ELM

It is noted that each Gaussian basis function in (3) contains two adjustable parameters, the center c_j and the width σ_j. The suitable parameters can help to improve the modeling performance. Compared to the conventional conjugate gradient and exhaustive search methods [25], the extreme learning machine (ELM) [9]-[11] assign random values to these parameters and is claimed to produce better generalization performance at a much faster learning speed and with least human intervene. Here, the ELM approach is employed to determine these parameters. It should be noted that the idea of choosing arbitrary data samples as the centers of candidate RBF hidden

node have been proposed early in [26]. The ELM further extended the idea by assigning randomly values to the nonlinear parameters, and Huang et al [10] further provided a theoretic framework to justify the efficacy of such approache.

The ELM [9]-[11] works for the generalized single-hidden layer feedforward networks (SLFNs). The essence of the ELM is that the nonlinear parameters in the hidden layer of SLFNs need not be tuned. From the interpolation capability point of view, it has been proved that if the activation function (i.e., nonlinear basis function) $\varphi_j(\cdot)$ is infinitely differentiable in any interval, the hidden layer parameters (i.e., c_j, σ_j) can be randomly generated. The corresponding theorem is as follows.

Theorem 1 [10] Given any small positive value $\varepsilon > 0$ and activation function $\varphi(\cdot)$ $g : R \to R$ which is infinitely differentiable in any interval, and N arbitrary distinct sample $(x(t), y(t)) \in \Re^m \times \Re^g$, there exists $M \leq N$ such that for any $\{c_j, \sigma_j\}_{j=1}^{M}$ randomly generated from any intervals of $\Re^m \times \Re$, according to any continuous probability distribution, then with probability one, the regression matrix Φ is invertible and $\|\Phi W - Y\| < \varepsilon$.

According to Theorem 1, the main idea behind an ELM is to randomly choose the centers and the widths. Using this idea, the following two steps are involved in forming the regression matrix Φ in (4). Firstly, the RBF centers c_j are determined by randomly selecting $M(M \leq N)$ samples from the training data. Secondly, the widths σ_j are found by generating random values uniformly within a specific range $[\sigma_{min}, \sigma_{max}]$. After the centers and the widths have been determined, the regression model (4) can be built using the training samples. The output weights can then be estimated using the following Theorem 2.

Theorem 2 (p. 51 of [27]) Let there exist a matrix G such that GY is a minimum norm least-squares solution of a linear system $\Phi W = Y$. Then it is necessary and sufficient that $G = \Phi^+$, the Moore-Penrose generalized inverse of matrix Φ.

2.3. Parameter estimation of multi-output linear-in-the-parameters model

To avoid over-fitting and obtain a sparse model, the regularization technique [19], [20] which introduces regularizers into the cost function is used. The regularization parameters can control the trade-off between the degree of regularization of the solution and its closeness to the data. A regularized cost function is introduced

$$J = tr(\Xi^T \Xi + \Theta^T \Lambda \Theta), \tag{5}$$

where $tr(\cdot)$ is the trace of matrix, $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_M)$ is a diagonal matrix, and $\lambda_i, i = 1, \ldots, M$, is the regularization parameter.

The regularized least-squares estimation to minimize (5) is given by

$$\hat{\Theta} = (\Phi^T\Phi + \Lambda)^{-1}\Phi^T Y. \tag{6}$$

if Φ is of full column rank.

Using (4), yields,

$$Y^TY - 2\hat{\Theta}^T\Lambda\hat{\Theta} = \hat{\Theta}^T\Phi^T\Phi\hat{\Theta} + \Xi^T\Phi\hat{\Theta} + \hat{\Theta}^T\Phi^T\Xi + \Xi^T\Xi - 2\hat{\Theta}^T\Lambda\hat{\Theta}. \tag{7}$$

Noting (6)

$$\begin{aligned}\hat{\Theta}^T\Phi^T\Xi - \hat{\Theta}^T\Lambda\hat{\Theta} &= \hat{\Theta}^T\Phi^T(Y - \Phi\hat{\Theta}) - \hat{\Theta}^T\Lambda\hat{\Theta} \\ &= \hat{\Theta}^T(\Phi^TY - \Phi^T\Phi\hat{\Theta} - \Lambda\hat{\Theta}) = 0.\end{aligned} \tag{8}$$

Similarly,

$$\Xi^T\Phi\hat{\Theta} - \hat{\Theta}^T\Lambda\hat{\Theta} = 0. \tag{9}$$

Equation (7) can be expressed as

$$\Xi^T\Xi + \hat{\Theta}^T\Lambda\hat{\Theta} = Y^TY - \hat{\Theta}^T(\Phi^T\Phi + \Lambda)\hat{\Theta}. \tag{10}$$

Thus, using (7)-(10), the associate minimum cost function for $\hat{\Theta}$ is calculated as

$$\begin{aligned}J(\hat{\Theta}) &= tr(\Xi^T\Xi + \hat{\Theta}^T\Lambda\hat{\Theta}) \\ &= tr((Y - \Phi\hat{\Theta})^T(Y - \Phi\hat{\Theta}) + \hat{\Theta}^T\Lambda\hat{\Theta}) \\ &= tr(Y^T(I - \Phi(\Phi^T\Phi + \Lambda)^{-1}\Phi^T)Y).\end{aligned} \tag{11}$$

3. Multi-output two-stage locally regularized model construction method

3.1. The net error reduction to the regularized cost function

In the forward subset selection procedure, suppose that k out of M candidate model terms/regressors, denoted as $p_1, \ldots, p_k$, have already been selected. The remaining regressors from the full regression matrix Φ are denoted as $\phi_{k+1}, \ldots, \phi_M$. The corresponding regularization parameters are also chosen in terms of the index label and shifted to the first k diagonal

elements of Λ becoming $\lambda_1, \ldots, \lambda_k$, and the remaining regularization parameters are denoted as $\lambda_{k+1}, \ldots, \lambda_M$. Therefore, the resulting regression matrix and regularization parameter matrix become $\Phi_k = [p_1, \ldots, p_k]$ and $\Lambda_k = diag\{\lambda_1, \ldots, \lambda_k\}$, respectively. According to (6) and (11), for an multi-output model with k regressors, it follows that

$$\hat{\Theta}_k = (\Phi_k^T \Phi_k + \Lambda_k)^{-1} \Phi_k^T Y, \tag{12}$$

$$J(\hat{\Theta}_k) = tr(Y^T (I - \Phi_k (\Phi_k^T \Phi_k + \Lambda_k)^{-1} \Phi_k^T) Y). \tag{13}$$

If a new $\forall \phi_j \in \{\phi_{k+1}, \ldots, \phi_M\}$ is now chosen, the corresponding regularization parameter $\Lambda_j \in \{\Lambda_{k+1}, \ldots, \Lambda_M\}$ is also selected in terms of the index label for ϕ_j. The regularized cost function is updated to

$$J(\hat{\Theta}_{k+1}) = tr(Y^T (I - \Phi_{k+1} (\Phi_{k+1}^T \Phi_{k+1} + \Lambda_{k+1})^{-1} \Phi_{k+1}^T) Y), \tag{14}$$

where $\Phi_{k+1} = [\Phi_k, \phi_j]$, $\Lambda_{k+1} = [\Lambda_k, \lambda_j]$ and the net error reduction of ϕ_j as the $(k+1)^{th}$ regressor is given by

$$\Delta J_{k+1}(\phi_j) = J(\hat{\Theta}_k) - J(\hat{\Theta}_{k+1}). \tag{15}$$

To select a new model term, the net error reduction (15) has to be calculated for each of the $M - k$ remaining candidate regressors (i.e., $\forall \phi_j \in \{\phi_{k+1}, \ldots, \phi_M\}$). The one that produces the largest error reduction to the regularized cost function is chosen as the $(k+1)^{th}$ model term, i.e.,

$$\Delta J_{k+1}(p_{k+1}) = \max\{\Delta J_{k+1}(\phi_j),\ \phi_j \in \{\phi_{k+1}, \ldots, \phi_M\}\}. \tag{16}$$

In this way, the multi-output model is constructed in a forward selection fashion. The model term selection continues until some model termination criterion is satisfied.

3.2. Stage 1-forward model selection

The above has presented the main points of the forward selection procedure based on the contribution of the candidate term to maximize the net model error reduction. However, to obtain the net error reduction of a new candidate term, $J(\hat{\Theta}_j)$ for each remaining candidate term must be computed using (14). Unfortunately, this requires extensive computational

effort. To facilitate numerical implementation and simplify the computational complexity, a regression context is established which includes several intermediate matrices that are generated during the model selection, i.e., the following defined matrices R_k, A, B, C and D.

First, define a matrix series R_k

$$R_k \triangleq \begin{cases} I - \Phi_k(\Phi_k^T\Phi_k + \Lambda_k)\Phi_k^T, & 1 \le k \le M, \\ \mathbf{I}, & k = 0. \end{cases} \tag{17}$$

The above defined matrix R_k has the following properties:

1)

$$R_{k+1} = R_k - \frac{R_k p_{k+1} p_{k+1}^T R_k}{p_{k+1}^T R_k p_{k+1} + \lambda_{k+1}}, k = 0, \ldots, M-1. \tag{18}$$

2)

$$R_k = R_k^T, k = 0, \ldots, M. \tag{19}$$

3)

$$R_{1,\ldots,p,\ldots,q,\ldots,k} = R_{1,\ldots,q,\ldots,p,\ldots,k}, \ p, q \le k. \tag{20}$$

Then define the following quantities, for arbitrary $N \times 1$ column vector which can be a selected regressor p_i or a remaining regressor ϕ, and the desired output matrix Y

$$p_i^{(k)} = R_k p_i, \ 1 \le i \le k; \quad \phi_j^{(k)} = R_k \phi_j, \ k < j \le M; \ Y^{(k)} = R_k Y, \tag{21}$$

where $Y^{(k)} = [y_1^{(k)}, \ldots, y_g^{(k)}]$ with $y_i^{(k)} = R_k y_i, 1 \le i \le g$, $p_i^{(0)} = p$, $\phi_j^{(0)} = \phi_j$ and $Y^{(0)} = Y$.

Using the above defined matrix series R_k and the quantities $p_i^{(k)}$, $\phi_j^{(k)}$ and $Y^{(k)}$, an intermediate $k \times M$ upper triangular matrix A can be defined as

$$\begin{aligned} &A \triangleq [a_{i,j}]_{k \times M}, \\ &a_{i,j} \triangleq \begin{cases} p_i^T R_{i-1} p_j = \left(p_i^{(i-1)}\right)^T p_j & i \le j \le k, \\ p_i^T R_{i-1} \phi_j = \left(p_i^{(i-1)}\right)^T \phi_j & k < j \le M. \end{cases} \end{aligned} \tag{22}$$

where k increases by one each time as a new model term is selected.

A lower triangular matrix C with zero diagonal elements is then defined as

$$\begin{aligned} &C \triangleq [c_{i,j}]_{k\times k} \\ &c_{i,j} \triangleq \begin{cases} 0 & i \le j, \\ p_i^T R_{i-1} p_j / \lambda_j = a_{i,j}/\lambda_j & j < i \le k. \end{cases} \end{aligned} \tag{23}$$

It is noted that the j^{th} column of the lower triangular in the matrix A is divided by the corresponding λ_j, becoming the j^{th} column in the matrix C.

A $M \times g$ matrix B is defined as

$$\begin{aligned} &B \triangleq [b_1, \ldots, b_i, \ldots, b_M]^T_{M\times g}, \\ &b_i = [b_i(1), \ldots, b_i(l), \ldots, b_i(g)], \quad 1 \le i \le M,\ 1 \le l \le g, \\ &b_i(l) \triangleq \begin{cases} p_i^T R_{i-1} y_l = p_i^T y_l - \sum_{s=1}^{i-1} \frac{a_{s,i} b_s(l)}{a_{s,s} + \lambda_s}, 1 \le i \le k, 1 \le l \le g, \\ \phi_i^T R_k y_l = \phi_i^T y_l - \sum_{s=1}^{i-1} \frac{a_{s,i} b_s(l)}{a_{s,s} + \lambda_s}, k \le i \le M, 1 \le l \le g. \end{cases} \end{aligned} \tag{24}$$

and the matrix D with k rows and $(k \times g)$ columns is also defined as

$$\begin{aligned} &D \triangleq [d_{i,j}]_{k\times(k\times g)}, \\ &d_{i,j} = [d_{i,j}(1), \ldots, d_{i,j}(l), \ldots, d_{i,j}(g)], \\ &d_{i,j}(l) \triangleq \begin{cases} 0, j \le i, 1 \le l \le g, \\ \frac{p_i^T R_{j-1} y_l}{\lambda_i} = \frac{b_i(l)}{a_{i,i} + \lambda_i} - \sum_{s=i+1}^{j-1} \frac{c_{s,i} b_s(l)}{a_{s,s} + \lambda_s}, i < j \le k, 1 \le l \le g. \end{cases} \end{aligned} \tag{25}$$

Using the properties of the matrix R_k and the matrices defined above,

(15) can be computed as

$$\begin{aligned}\Delta J_{k+1}(\phi_j) &= J(\hat{\Theta}_k) - J(\hat{\Theta}_{k+1}) \\ &= tr(Y^T(I - \Phi_k(\Phi_k^T\Phi_k + \Lambda_k)^{-1}\Phi_k^T)Y) \\ &\quad - tr(Y^T(I - \Phi_{k+1}(\Phi_{k+1}^T\Phi_{k+1} + \Lambda_{k+1})^{-1}\Phi_{k+1}^T)Y) \\ &= tr(Y^T R_k Y) - tr(Y^T R_{k+1} Y) \\ &= tr\left(\frac{Y^T R_k \phi_j \phi_j^T R_k Y}{\phi_j^T R_k \phi_j + \lambda_j}\right) \\ &= tr\left(\frac{b_j^T b_j}{a_{j,j} + \lambda_j}\right) \\ &= \frac{b_j b_j^T}{a_{j,j} + \lambda_j}\end{aligned} \tag{26}$$

where b_j, $j = k+1, \ldots, M$, is the j^{th} row of the matrix B.

To further reduce the calculations involved in the model term selection procedure, at the $(k+1)^{th}$ step, $a_{j,j}^{k+1}$ and b_j^{k+1} $(j = k+1, \ldots, M)$ for $\forall \phi_j \in \{\phi_{k+1}, \ldots, \phi_M\}$ can be updated recursively as

$$\begin{aligned} a_{j,j}^{k+1} &= (\phi_j^{(k)})^T \phi_j = a_{j,j}^k - \frac{a_{k,j}^2}{a_{k,k} + \lambda_k}, \\ b_j^{k+1} &= [b_j^{k+1}(1), \ldots, b_j^{k+1}(l), \ldots, b_j^{k+1}(g)], \\ b_j^{k+1}(l) &= (\phi_j^{(k)})^T y_l = b_j^k(l) - \frac{a_{k,j} b_k(l)}{a_{k,k} + \lambda_k}, \end{aligned} \tag{27}$$

where $a_{j,j}^{k+1}$ and b_j^{k+1} denote the values at the $(k+1)^{th}$ step, $a_{j,j}^k$ and b_j^k denote the values at the k^{th} step. By default, $a_{j,j}^k$ and b_j^k will be denoted as $a_{j,j}$ and b_j in the following. Thus, at the end of each model term selection, these terms are updated and stored for use in the model term selection. After $a_{j,j}^{k+1}$ and b_j^{k+1} have been recursively updated using (27), $\Delta J_{k+1}(\phi_j)$ follows from (26). Thus, the net error reductions to the regularized cost function for each remaining candidate model term can be efficiently computed.

3.3. Stage 2-backward model refinement

The forward selection stage is subject to the constraint that all previously selected model terms remain fixed and can not be removed from the model

later. Unfortunately, these model terms introduced later may affect the contribution of previously selected ones. As a result, the previously selected model terms may become insignificant due to inclusion of later ones. To overcome this deficiency and construct a more compact multi-output model, each previously selected model term needs to be checked again and its net error reduction to the regularized cost function needs also to be compared with all remaining ones in the candidate pool. Any insignificant terms are replaced.

3.3.1. Regression context reconstruction

After an initial model with n model terms has been produced from the first stage, the selected n model terms are shifted in a group to the left hand side of Φ, the resulting regression matrix becoming $\Phi = [p_1, p_2, \ldots, p_n, \phi_{n+1}, \ldots, \phi_M]$. The corresponding regularization parameter is expressed as $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_n, \lambda_{n+1}, \ldots, \lambda_M]$. To review each previously selected regressor term, it must be shifted to the n^{th} position through a series of interchanges between adjacent regressor terms.

Suppose that p_q and p_{q+1} in the regression matrix Φ are interchanged, i.e.,

$$\hat{p}_q = p_{q+1}, \hat{p}_{q+1} = p_q, \;\; q = 1, \ldots, n-1.$$

This will make the regularization parameters and the above defined regression context to be changed as follows:

1. According to the index label of p_q and p_{q+1}, the corresponding regularization parameters are changed as

$$\hat{\lambda}_q = \lambda_{q+1}, \hat{\lambda}_{q+1} = \lambda_q.$$

2. According to the property given in (20), only R_q in the matrix series $R_k (k = 1, \ldots, n)$ is changed at each step, becoming $\hat{R}_q$. $\hat{R}_q$ can be recalculated as

$$\hat{R}_q = R_{q-1} - \frac{R_{q-1}\hat{p}_q(\hat{p}_q)^T R_{q-1}}{(\hat{p}_q)^T R_{q-1}\hat{p}_q + \hat{\lambda}_q}. \tag{28}$$

3. For the matrix A, the q^{th} and $(q+1)^{th}$ columns with elements from row 1 to $q-1$ are exchanged, i.e., $\hat{a}_{i,q} = a_{i,q+1}, \;\; \hat{a}_{i,q+1} = a_{i,q}, \;\; i = 1, \ldots, q-1$. The q^{th} and $(q+1)^{th}$ rows are then recalculated as

$$\hat{a}_{q,j} = \begin{cases} a_{q+1,q+1} + a_{q,q+1}^2/(a_{q,q} + \lambda_q) & j = q, \\ a_{q,q+1} & j = q+1, \\ a_{q+1,j} + (a_{q,q+1}a_{q,j})/(a_{q,q} + \lambda_q) & q+2 \le j \le M. \end{cases} \tag{29}$$

$$\hat{a}_{q+1,j} = \begin{cases} a_{q,q} - a_{q,q+1}^2 \Big/ (\hat{a}_{q,q} + \hat{\lambda}_q) & j = q+1, \\ a_{q,j} - a_{q,q+1}\hat{a}_{q,j} \Big/ (\hat{a}_{q,q} + \hat{\lambda}_q) & q+2 \le j \le M. \end{cases} \tag{30}$$

4. For the matrix C, the q^{th} and $(q+1)^{th}$ columns with elements from row $q+2$ to n are exchanged, i.e., $\hat{c}_{i,q} = c_{i,q+1}, \ \hat{a}_{i,q+1} = c_{i,q}, \quad i = q+2, \ldots, n.$ The q^{th} and $(q+1)^{th}$ rows are then given by

$$\hat{c}_{q,j} = c_{q+1,j} + a_{q,q+1}c_{q,j}/(a_{q,q} + \lambda_q), \quad 1 \le j \le q-1. \tag{31}$$

$$\hat{c}_{q+1,j} = \begin{cases} a_{q,q+1} \Big/ (\hat{a}_{q,q} + \hat{\lambda}_q) & j = q, \\ c_{q,j} - a_{q,q+1}\hat{c}_{q,j} \Big/ (\hat{a}_{q,q} + \hat{\lambda}_q) & 1 \le j < q. \end{cases} \tag{32}$$

5. For the matrix B, the q^{th} and $(q+1)^{th}$ row are altered

$$\begin{aligned} \hat{b}_q(l) &= b_{q+1}(l) + a_{q,q+1}b_q(l)/(a_{q,q} + \lambda_q) \ , \ l = 1, \ldots, g. \\ \hat{b}_{q+1}(l) &= b_q(l) - a_{q,q+1}\hat{b}_q(l) \Big/ \left(\hat{a}_{q,q} + \hat{\lambda}_q\right) \ , \ l = 1, \ldots, g. \end{aligned} \tag{33}$$

6. For the matrix D, the q^{th} and $(q+1)^{th}$ rows with elements from column $(q+1) \times g + 1$ to $n \times g$ are exchanged, i.e.,

$$\begin{aligned} \hat{d}_{q,j}(l) &= d_{q+1,j}(l) \ , \quad j = q+2, \ldots, n, \ l = 1, \ldots, g. \\ \hat{d}_{q+1,j}(l) &= d_{q,j}(l) \ , \quad j = q+2, \ldots, n, \ l = 1, \ldots, g. \end{aligned}$$

The $(q+1)^{th}$ column with elements from row 1 to q are changed to

$$\hat{d}_{i,q+1}(l) = \begin{cases} \hat{b}_q(l) \Big/ \left(\hat{a}_{q,q} + \hat{\lambda}_q\right) & i = q, \\ d_{i,q}(l) - \hat{b}_q(l)\hat{c}_{q,i} \Big/ \left(\hat{a}_{q,q} + \hat{\lambda}_q\right) & i < q. \end{cases} \tag{34}$$

The above procedure continues until the k^{th} model term is shifted to the n^{th} position, i.e., $\hat{p}_n = p_k$. The corresponding R_{n-1} is changed, becoming $\hat{R}_{n-1}$, but R_n is not changed. According to the definition of the matrix A in (22), $a_{j,j}^n$ and b_j^n for each remaining candidate terms are also changed, becoming $\hat{a}_{j,j}^n$ and $\hat{b}_j^n$. The $\hat{a}_{j,j}^n$ and $\hat{b}_j^n$ can be computed by

$$\begin{aligned} \hat{a}_{j,j}^n &= a_{j,j}^{n+1} + \frac{\hat{a}_{n,j}^2}{\hat{a}_{n,n} + \hat{\lambda}_n}, \\ \hat{b}_j^n &= [\hat{b}_j^n(1), \ldots, \hat{b}_j^n(l), \ldots, \hat{b}_j^n(g)], \\ \hat{b}_j^n(l) &= b_j^{n+1}(l) + \frac{\hat{a}_{n,j}\hat{b}_n(l)}{\hat{a}_{n,n} + \hat{\lambda}_n}. \end{aligned} \tag{35}$$

To review the significance of the shifted term p_k and those remaining candidate terms, their net error reduction to the regularized cost function need be recalculated again, i.e.,

$$\begin{aligned}\Delta\hat{J}_n(\hat{p}_n = p_k) &= tr\left(\frac{\hat{b}_n^T\hat{b}_n}{\hat{a}_{n,n}+\hat{\lambda}_n}\right) = \frac{\hat{b}_n\hat{b}_n^T}{\hat{a}_{n,n}+\hat{\lambda}_n},\\ \Delta\hat{J}_n(\phi_j) &= tr\left(\frac{\hat{b}_j^T\hat{b}_j}{\hat{a}_{j,j}+\hat{\lambda}_j}\right) = \frac{\hat{b}_j\hat{b}_j^T}{\hat{a}_{j,j}+\hat{\lambda}_j}.\end{aligned} \tag{36}$$

Assuming $\Delta\hat{J}_n(\phi_m) = \max\{\Delta\hat{J}_n(\phi_j),\ \phi_j \in \{\phi_{n+1}, \phi_{n+2}, \ldots, \phi_M\}\}$, and if $\Delta\hat{J}_n(\phi_m) > \Delta\hat{J}_n(\hat{p}_n = p_k)$, then ϕ_m will replace $\hat{p}_n$ in the selected regression matrix Φ_n. Meanwhile $\hat{p}_n$ will be put back into the candidate term pool and take the position of ϕ_m, the corresponding elements in the regression context must also be updated as follows.

3.3.2. Updating the regression context

1. According to the index label of $\hat{p}_n$ and ϕ_m, the n^{th} element and m^{th} element in the regularization parameters matrix Λ are exchanged, i.e., $\hat{\lambda}_n = \lambda_m$, $\hat{\lambda}_m = \lambda_n$.

2. For the matrix A, the n^{th} and m^{th} columns with elements from row 1 to $n-1$ are exchanged, i.e., $\hat{a}_{i,n} = a_{i,m}$, $\hat{a}_{i,m} = a_{i,n}$, $i = 1, \ldots, n-1$. The n^{th} row is changed to

$$\hat{a}_{n,j} = \begin{cases} a_{m,m} & j = n, \\ a_{n,m} & j = m, \\ \phi_m^T\phi_j - \sum_{s=1}^{n-1}(a_{s,m}a_{s,j})/(a_{s,s}+\lambda_s) & \forall j \neq n, j \neq m. \end{cases} \tag{37}$$

3. For the matrix C, the n^{th} row with elements from column 1 to $n-1$ are updated using

$$\hat{c}_{n,j} = \frac{\hat{a}_{j,n}}{(a_{j,j}+\lambda_j)} - \sum_{s=j+1}^{n-1}\frac{\hat{a}_{s,n}c_{s,j}}{(a_{s,s}+\lambda_s)}. \tag{38}$$

4. For the matrix B, the n^{th} row with elements from column 1 to g equals to the m^{th} row with elements from column 1 to g, i.e., $\hat{b}_n(l) = b_m(l), \quad l = 1, \ldots, g$.

5. Finally, because the n^{th} and m^{th} model terms are exchanged, the corresponding R_n is changed, becoming $\hat{R}_n$. According to the recursive computation formulas of $a_{j,j}^{n+1}$ and b_j^{n+1} in (27), $a_{j,j}^{n+1}$ and b_j^{n+1} for each remaining candidate terms are also changed, becoming $\hat{a}_{j,j}^{n+1}$ and $\hat{b}_j^{n+1}$. The $\hat{a}_{j,j}^{n+1}$ and $\hat{b}_j^{n+1}$ can be altered

$$\hat{a}_{j,j}^{(n+1)} = \begin{cases} a_{n,n} - \left(\hat{a}_{n,m}\right)^2 \Big/ \left(\hat{a}_{n,n} + \hat{\lambda}_n\right) & j = m, \\ \hat{a}_{j,j} - \left(\hat{a}_{n,j}\right)^2 \Big/ \left(\hat{a}_{n,n} + \hat{\lambda}_n\right) & j \neq m,\ n < j \leq M. \end{cases} \tag{39}$$

$$\hat{b}_j^{(n+1)}(l) = \begin{cases} b_n(l) - \hat{a}_{n,m}\hat{b}_n(l) \Big/ \left(\hat{a}_{n,n} + \hat{\lambda}_n\right) & j = m, \\ b_j(l) - \hat{a}_{n,j}\hat{b}_n(l) \Big/ \left(\hat{a}_{n,n} + \hat{\lambda}_n\right) & j \neq m,\ n < j \leq M. \end{cases} \tag{40}$$
$$l = 1, \ldots, g.$$

3.3.3. Computation of the regression coefficients

The regression coefficients with n model terms can be easily derived using the regression context. It follows from the definition of R_k that

$$\Phi_k^T R_k = \Lambda_k (\Phi_k^T \Phi_k + \Lambda_k)^{-1} \Phi_k^T. \tag{41}$$

Referring to (12) and using (41), (12) can be revised as

$$\hat{\Theta}_k = (\Phi_k^T \Phi_k + \Lambda_k)^{-1} \Phi_k^T Y = \Lambda_k^{-1} \Phi_k^T R_k Y. \tag{42}$$

According to the definition of the matrix D in (25), the above coefficients $\hat{\Theta}_k$ can be easily solved by

$$\hat{\theta}_{i,l} = d_{i,k+1}(l) = \frac{b_i(l)}{a_{i,i} + \lambda_i} \sum_{s=i+1}^{k} \frac{c_{s,i} b_s(l)}{a_{s,s} + \lambda_s}, \quad i = 1, \ldots, k, \ \ l = 1, \ldots, g. \tag{43}$$

3.3.4. Updating the regularization parameter

When the multi-output model refinement procedure is completed, the regularization parameters can be optimized by the Bayesian evidence procedure [19], [20]. Hence, the regularization parameters are updated by

$$\lambda_i^{new} = \frac{\gamma_i}{N - \gamma} \frac{\sum_{l=1}^{g} \varepsilon_l^T \varepsilon_l}{\sum_{l=1}^{g} \theta_{i,l}^2}, \ \ 1 \leq i \leq n$$
$$\gamma = \sum_{i=1}^{M} \gamma_i, \quad \gamma_i = 1 - \lambda_i v_i, \tag{44}$$

where v_i is the i^{th} diagonal element of the inverse of the matrix $(\Phi_n^T\Phi_n + \Lambda_n)$. Finding a (local) optimal regularization parameters, usually requires a few iterations (typically after 10 iterations) [23]. Define $V_k = \left[\ v_1,\ \ \ldots, v_i, \ldots\ \ v_k\ \right] = diag((\Phi_k^T\Phi_k + \Lambda_k)^{-1})$. The vector V_k can be updated recursively as

$$V_{k+1} = \left[\ V_k + [c_{k+1}]^{.2}(a_{k+1,k+1} + \lambda_{k+1})^{-1},\ \ (a_{k+1,k+1} + \lambda_{k+1})^{-1}\ \right],$$

where c_{k+1} is the $k+1^{th}$ row with elements from column 1 to k of the matrix C, and $[\cdot]^{.2}$ denotes the square operation on each element.

3.4. Complete algorithm

The multi-output two-stage locally regularized model construction method using the extreme learning machine can now be summarized as follows.

Step 1 Initialization:

(A) After the data samples are collected and the center parameters for the Gaussian functions being randomly selected from the training data and the widths being randomly selected from a specific range $[\sigma_{min}, \sigma_{max}]$ using the concept of the ELM, the candidate regression matrix Φ are constructed.

(B) Set λ_i, $1 \leq i \leq M$, to the same small positive value (e.g., 10^{-6}), and the iteration index $I = 1$.

Step 2 Forward selection:

(A) Set the model size $k = 0$. At the first step, calculate $a_{j,j}$ and b_j using (22). The net error reductions to the regularized cost function are then computed using (26), and the model term that produced the largest error reduction is chosen. The $a_{j,j}^2$ and b_j^2 are updated for the next selection.

(B) At the k^{th} step, for $\forall\phi_i \in \{\phi_k, \ldots, \phi_M\}$, the net error reduction to the regularized cost function in (26) is calculated using the $a_{j,j}^k$ and b_j^k, respectively. Find the k^{th} model term using (16) and add it to the regression matrix $\Phi_k = [\Phi_{k-1}, p_k]$. Update $a_{j,j}^{k+1}$ and b_j^{k+1} using (27).

(C) The procedure is terminated when some criteria is satisfied, which produces a $(k-1)$-unit model. Otherwise, set $k = k+1$, and go to step 2 (A).

Step 3 Backward model refinement:

(A) Interchange the positions of p_k and p_{k+1} ($k = n-1, \ldots, 1$), and update the related terms according to (28)-(34). This process continues until the regressor term p_k is moved to the n^{th} position.

(B) Update the $\hat{a}^n_{j,j}$ and $\hat{b}^n_j$ using (35), and compute their new net error reduction to the regularized cost function. If $\Delta\hat{J}_n(\phi_m) > \Delta\hat{J}_n(\hat{p}_n = p_k)$, then ϕ_m will replace $\hat{p}_n$ in the selected regression matrix Φ_n. Meanwhile $\hat{p}_n$ will be put back into the candidate term pool and take the position of ϕ_m.

(C) Update the related terms in the regression context according to (37)-(40). If $k > 1$, set $k = k - 1$, and go to step 3 (A).

(D) If one or more regressor terms were changed, then set $k = n - 1$, and repeat steps 3(A)-3(C) to review all the terms again.

Step 4 Updating the regularization parameters:

(A) Using the final set of selected regressor terms to calculate the model coefficient vector $\hat{\Theta}$ and update the regularization parameters Λ using (44).

(B) If λ remains largely unchanged in two successive iterations, or a pre-set maximum iteration number is reached, stop; otherwise, update the candidate set by $k - 1$ selected significant centers, set $I = I + 1$, and go to step 2.

3.5. Computational complexity analysis

The computation time mainly arises from multiplication/division operations, only these values are counted here. The computation involved in the proposed algorithm is dominated by the selection of g output LITP model terms. Suppose there are initially M candidate terms, from which only n terms are eventually selected ($n \leq M$) when N data samples are used for modeling.

The complexities involving multiplications/divisions of MOLS and MFRA are first compared to the proposed method. The total number of multiplication/division operations can be calculated respectively as [16]

$$\begin{aligned} S_{MOLS} &\approx (2g+2)nMN, \\ S_{MFRA} &\approx nMN. \end{aligned} \tag{45}$$

For the proposed method, the computational expense involving multiplicaiton/division operations consists two parts, i.e., stage 1 and stage 2.

Stage 1: Suppose there are M candidate model terms with N data samples being used for training. The computational expense involving multiplication/division operations is dominated by the computation of the regression context, i.e., the matrices A, B, C and D. If a g output LITP model with n regressor terms is constructed, the number of multiplication/division operations is given by

$$S_{1st} \approx (2g+3)nM + nNM \tag{46}$$

Stage 2: The computational expense involving multiplication/division operations is dominated by the computation of the regression context reconstruction and updating the regularization parameters. The number of multiplication/division operations for one complete check loop in the worst case is calculated as

$$S_{2nd} \approx NM + (n^2+g)M - nN \tag{47}$$

However, if more than one check loops, e.g. L check loops were performed, then the total computational expense will be less than $L \times S_{2nd}$. In practice, $n \ll N$ and $n \ll M$, if I (typically after 10 iterations) [23] is the number of iterations in updating the regularization parameters, the total number of multiplication/division operations then becomes

$$S_{MTLRMC}^{total} \approx (n+L)\, INM \tag{48}$$

The ratios of (48) and (45) are

$$\begin{aligned} S_{MTLRMC/MOLS}^{Ratio} &= \frac{(n+L)I}{(2g+2)n} \times 100\%, \\ S_{MTLRMC/MFRA}^{Ratio} &= \frac{(n+L)I}{n} \times 100\%, \end{aligned} \tag{49}$$

and provide a convenient index for comparing the computations of three methods. Figs.1 and 2 shows the ratio with varying model size (n) and different number of iterations in updating the regularization parameters (I). It shows that the computation involved in the proposed MTLRMC method is larger than the MOLS and MFRA methods.

4. Simulation examples

The proposed algorithm was applied to two different problems. All the numerical simulation were carried out using MATLAB on a PIV-2.27-GHZ personal computer (PC) with windows 7.

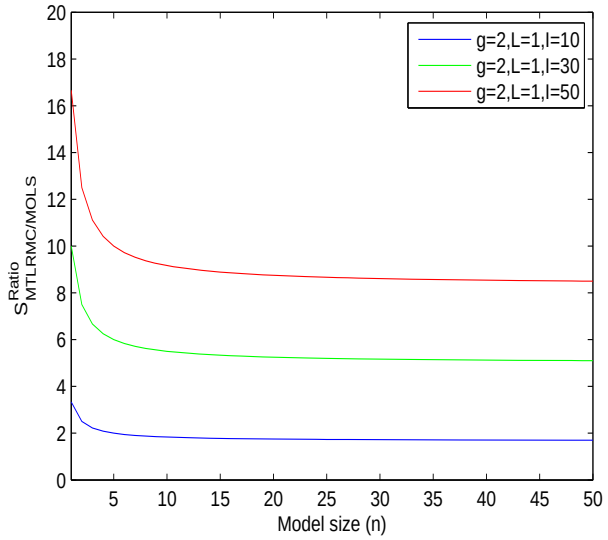


Figure 1: Comparison of the computations of $MTLRMC/MOLS$

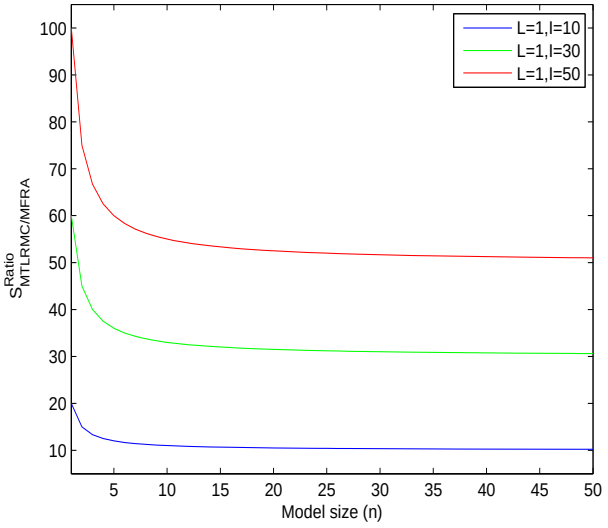


Figure 2: Comparison of the computations of $MTLRMC/MFRA$

4.1. Example 1: two-output nonlinear system

Consider the following benchmark two-output nonlinear system [24]:

$$\begin{aligned} y_1(k) &= 0.5y_1(k-1) + u(k-1) + 0.4\tanh(u(k-2)) \\ &\quad + 0.1\sin(\pi y_1(k-2))y_2(k-1) + \varepsilon_1(k) \\ y_2(k) &= 0.3y_2(k-1) + 0.1y_2(k-2)y_1(k-1) \\ &\quad + 0.4\exp(-u^2(k-1))y_1(k-2) + \varepsilon_2(k), \end{aligned} \tag{50}$$

where the system input $u(k)$ was uniformly distributed in $(-0.5, 0.5)$, and $\mathrm{E} = [\varepsilon_1(k), \varepsilon_2(k)]^T$ were the zero-mean Gaussian white noise with covariance δI_2.

Initial conditions were set as $y_1(0) = y_1(-1) = y_2(0) = y_2(-1) = 0$, $u(0) = u(-1) = 0$, and 500 training data points were generated by simulating (50) with noise covariance $0.3I_2$. Another 500 noise-free data points were generated for testing. Following [24], $y_1(k-1)$, $y_1(k-2)$, $y_2(k-1)$, $y_2(k-2)$, $u(k-1)$, $u(k-2)$ were used as the multi-output RBF model inputs. In the experiment, $\varphi(x, c_i) = \exp(-\|x - c_i\|/\sigma_i^2)$ was used as the basis function for the RBF neural model.

The proposed MTLRMC method was used to construct multi-output RBF model. Using the concept of the ELM, the centers $(c_i, 1 \le i \le M)$ were obtained by randomly selecting samples from the training data, and the widths $(\delta_i, 1 \le i \le M)$ were randomly selected from a specific range $[0.9, 1.2]$. The selection process was performed as follows. After the first iteration of the regularization parameters, an initial multi-output RBF model with 26 hidden nodes was produced. The neural model was then refined until the regularization parameter vector λ converged at the 10^{th} iteration. Some values of the regularization parameters and weights are listed in Table 1. It is shown that all hidden nodes added after the 24^{th} have very large regularization parameters and their corresponding weights are very small. Therefore, the final multi-output RBF neural model has only 24 hidden nodes, with a reduction of about 8% of the hidden nodes compared to alternatives. It is indicated that the proposed algorithm can remove redundant hidden nodes.

The MFRA, MOLS and ELM methods were also used to construct multi-output RBF neural model respectively. For the MOLS approach, the model construction procedure is terminated using the Akaike's information criterion (AIC) [28]. In this way, a multi-output RBF model with 26 hidden nodes was constructed. For the MFRA, a similar multi-output RBF neural model with 26 hidden nodes was also generated but with much less computational effort.

Table 1: Model weights and regularisation parameters (Set $\lambda_i = 1 \times 10^{-4}, 1 \leq i \leq 500$)

Model term l	Model weight $\theta_{l,1}$	Model weight $\theta_{l,2}$	regulariser λ_l
1	-3.5203	-0.2069	0.1816
2	3.0948	-0.2218	0.2158
3	-0.8721	1.7127	0.5091
4	3.7444	0.1947	0.1636
5	-2.1169	-1.2629	0.3401
⋮	⋮	⋮	⋮
22	-0.1535	-0.1536	6.5345
23	0.0638	-0.0142	16.4576
24	-0.0237	0.0291	71.6858
25	-4.300×10^{-3}	1.430×10^{-3}	6.756×10^{2}
26	4.700×10^{-3}	3.407×10^{-4}	5.817×10^{2}

For the ELM algorithm, 26 hidden nodes were used, and the corresponding multi-output RBF model was produced.

The final results of these four algorithms, including model size, the execution time, training error covariance ($Cov(E_{Tr})$) and testing error covariance ($Cov(E_{Te})$) are summarized in Table 2. The generalization capability of an identified model can best be tested by examining the iterative model output, so the iterative model error covariance ($Cov(E_I)$) is also listed in Table 2 where the smallest model size and best $Cov(E_I)$ were underlined. With regard to the computational complexity, the execution time of the proposed algorithm is longer than the other three methods as expected. This is largely due to the computational expense of the iterative optimization of the regularization parameters (λ) for the 26 candidate nodes at the first iteration. Compared to the other three methods, the proposed new algorithm was able to produce a sparser multi-output RBF model while achieve better one-step-ahead prediction and generalization capability.

To further test the generalization performance and effectiveness of the proposed method, different levels of noise were added to the system output. The forward selection procedure is used to construct multi-output models in the MOLS and MFRA methods. The ELM produces multi-output models with the preset number of model terms. Unlike these three methods, the pro-

Table 2: Performance of multi-output RBF neural model constructed by four algorithms

Algorithm	Size (m)	Time (sec.)	Training $Cov(E_{Tr})$		Testing $Cov(E_{Te})$		Recurrent $Cov(E_I)$	
MFRA	26	0.67	124.82	-1.99	7.99	2.90	16.49	10.67
			-1.99	119.49	2.90	3.45	10.67	13.05
MOLS	26	2.29	123.21	-0.94	7.64	3.14	17.77	12.63
			-0.94	119.64	3.14	4.39	12.63	14.48
ELM	26	0.62	137.19	2.38	12.72	4.78	26.94	18.91
			2.38	139.73	4.78	7.11	18.91	24.25
MTLRMC	24	4.76	130.58	-5.98	6.61	2.45	11.96	6.43
			-5.98	122.85	2.45	4.14	6.43	12.34

posed MTLRMC method firstly determines the nonlinear parameters in each model term by using the concept of the ELM. At the first iteration of regularization parameters optimization, an initial multi-output LITP model was generated by forward selection and backward model refinement. In the subsequent regularization parameters optimization stage, the associated weights of those nonsignificant candidate hidden nodes were effectively forced to zero as their corresponding regularization parameters became sufficiently large. Therefore, a sparser model with better generalization capability can be produced with the aid of local regularization technique. Table 3 shows the test results, where the smallest model size and best $Cov(E_I)$ were underlined. The computational complexity of the proposed algorithm is greater than that of all other three algorithms. However, the proposed algorithm again produced better generalization performance with much less model terms.

4.2. Example 2: two-output time-series

Consider the following two-output time-series [13], [24]:

$$\begin{aligned} y_1(k) &= 0.1\sin(\pi y_2(k-1)) \\ &\quad + (0.8 - 0.5\exp(-y_1^2(k-1)))y_1(k-1) \\ &\quad - (0.3 + 0.9\exp(-y_1^2(k-1)))y_1(k-2) + \varepsilon_1(k) \\ y_2(k) &= 0.6y_2(k-1) + 0.2y_2(k-1)y_2(k-2) \\ &\quad + 1.2\tanh(y_1(k-2)) + \varepsilon_2(k), \end{aligned} \tag{51}$$

Table 3: Performance of multi-output RBF constructed by four algorithms with different noise levels

Noise		0.4		0.5		0.6	
MOLS	Size	21		23		30	
	Time (s)	2.2183		2.6208		3.8220	
	Training	100.25	1.49	98.74	-2.20	95.73	-5.10
	$Cov(E_{Tr})$	1.49	97.80	-2.20	99.08	-5.10	97.30
	Testing	4.80	1.26	6.80	0.03	3.69	0.24
	$Cov(E_{Te})$	1.26	2.07	0.03	1.41	0.24	0.92
	Recurrent	14.98	9.06	12.20	2.69	10.23	3.44
	$Cov(E_I)$	9.06	9.63	2.69	5.53	3.44	3.78
MFRA	Size	21		23		30	
	Time (s)	0.4680		0.6396		0.7956	
	Training	99.73	1.15	97.98	-2.65	95.22	-3.38
	$Cov(E_{Tr})$	1.15	97.04	-2.65	98.85	-3.38	97.69
	Testing	4.07	1.19	4.64	0.25	3.16	-0.13
	$Cov(E_{Te})$	1.19	2.40	0.25	2.13	-0.13	1.34
	Recurrent	13.33	8.04	8.75	2.68	9.42	3.62
	$Cov(E_I)$	8.04	9.25	2.68	5.93	3.62	3.88
ELM	Size	21		23		30	
	Time (s)	0.0312		0.0343		0.0936	
	Training	133.75	6.82	131.14	-2.06	101.71	-2.70
	$Cov(E_{Tr})$	6.82	108.26	-2.06	110.07	-2.70	104.74
	Testing	12.28	5.39	11.74	2.53	3.84	1.85
	$Cov(E_{Te})$	5.39	5.15	2.53	4.18	1.85	4.04
	Recurrent	62.13	44.30	19.45	6.97	9.79	6.89
	$Cov(E_I)$	44.30	41.41	6.97	14.69	6.89	12.10
MTLRMC	Size	20		22		28	
	Time (s)	2.9216		4.3992		5.3820	
	Training	101.13	0.31	106.48	-1.47	98.14	-5.27
	$Cov(E_{Tr})$	0.31	99.25	-1.47	101.30	-5.27	101.14
	Testing	4.19	0.53	2.18	-0.41	2.11	-0.57
	$Cov(E_{Te})$	0.53	1.46	-0.41	1.23	-0.57	1.82
	Recurrent	10.40	4.66	3.88	-0.23	5.78	1.61
	$Cov(E_I)$	4.66	6.54	-0.23	2.57	1.61	3.34

where E $= [\varepsilon_1(k), \varepsilon_2(k)]^T$ were the zero-mean Gaussian white noise with covariance $0.04I_2$. Initial conditions were set as $y_1(0) = y_1(-1) = y_2(0) = y_2(-1) = 0$, and 1000 noisy measurements were generated to construct the multi-output RBF neural model by simulating (51). The inputs to the RBF model were chosen as $y_1(k-1)$, $y_1(k-2)$, $y_2(k-1)$ and $y_2(k-2)$.

Unlike the previous study [16], the MFRA method constructed a multi-output RBF model with 37 hidden nodes for the given training data with the noise covariance $0.01I_2$. Here, the noise level was much higher.

The MFRA, MOLS and ELM methods were used to construct the multi-output RBF neural models, respectively. Akaike's information criterion (AIC) [28] was used as the termination criteria for the MFRA and MOLS again. The MFRA and MOLS produced an multi-output RBF model with 95 and 66 hidden nodes, respectively. For the ELM algorithm, 26 hidden nodes were set, and the corresponding multi-output RBF neural model was produced.

The proposed new algorithm was applied to the same dataset. Using the concept of the ELM, the centers $(c_i, 1 \leq i \leq 1000)$ were obtained by randomly selecting samples from the training data, and the widths $(\delta_i, 1 \leq i \leq 1000)$ were randomly selected from a specific range $[0.9, 1.5]$. The final multi-output RBF neural model with only 26 hidden nodes was produced.

The final results of the four algorithms, including model size, training error covariance ($Cov(E_{Tr})$) and testing error covariance ($Cov(E_{Te})$) are summarized in Table 4 where the best $Cov(E_Te)$ were underlined. Compared to the other three methods, simulation results confirm that the proposed algorithm was able to produce a sparser multi-output RBF model while achieve better generalization capability again.

5. Concluding remarks

A novel multi-output two-stage locally regularized model construction method using the extreme learning machine has been proposed. In this algorithm, the nonlinear parameters in each model term are firstly determined by using the concept of the ELM. An initial multi-output LITP model is generated according to the termination criteria from the first stage. The significance of each selected regressor is then checked and insignificant ones are replaced at the second stage. The proposed method can produce an optimized compact model by the regularization parameters. Further, to reduce the computational complexity, a proper regression context has been used to allow fast implementation of the proposed method. Simulation results show

Table 4: Performance of multi-output RBF neural model constructed by four algorithms

Algorithm	Size (m)	Training $Cov(E_{Tr})$		Testing $Cov(E_{Te})$	
MFRA	95	33.8042	-0.9921	0.5454	0.0459
		-0.9921	35.5659	0.0459	0.4324
MOLS	66	35.2295	-0.7396	0.4001	0.0708
		-0.7396	37.4450	0.0708	0.7497
ELM	26	75.0107	-26.0092	6.7753	-3.7752
		-26.0092	80.2866	-3.7752	5.3221
MTLRMC	26	40.3011	1.0173	0.4036	-0.0113
		1.0173	39.8990	-0.0113	0.1583

that the computational complexity of the proposed algorithm is greater than that of alternative methods. However, the proposed method can produce a sparser multi-output model while achieve better generalization capability.

Acknowledgment

This work was supported in part by the Research Councils UK under grant EP/G042594/1, the National Science Foundation of China (61074032, 51007052, 61104089), Science and Technology Commission of Shanghai Municipality (11ZR1413100), Leading Academic Discipline Project MEE&AMA of Shanghai University, the innovation fund project for Shanghai University.

[1] C.-N. Ko, Integration of support vector regression and annealing dynamical learning algorithm for MIMO system identification. Expert Systems with Applications. 38(2011) 15224-15233.

[2] B. S. Dayal and J. F. MacGregor. Multi-output process identification. Journal Process Control. 7(4) (1997) 269-282.

[3] L. L. Han and F. Ding. Multi-innovation stochastic gradient algorithms for multi-input multi-output systems. Digital Signal Process. 19(4) (2009) 545-554.

[4] L. L. Xiang, L. B. Xie, Y. W. Liao and R. F. Ding. Hierarchical least squares algorithms for single-input multiple-output systems based on

the auxiliary model. Mathematical and Computer Modelling, 52(6) (2010) 918-924.

[5] F. Ding, Y. Liu and B. Bao. Gradient-based and least-squares-based iterative estimation algorithms for multi-input multi-output systems. Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering. 226(1) (2012) 43-55.

[6] J. Barry Gomm, D. Yu, Selection radial basis function network centers with recursive orthogonal least squares training, IEEE Trans. Neural Netw. 11 (2000), 306-314.

[7] S. Elanayar, Y. C. Shin. Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. IEEE Trans. Neural Netw. 5(4) (1994) 594-603.

[8] L. Xu, A. Krzyzak and A. Yuille. On radial basis function nets and kernel regression: statistical consistency, convergence rates, and receptive field size, Neural Networks. 7(4) (1994) 609-628.

[9] G.-B. Huang, D. H. Wang and Y. Lan. Extreme learning machines: a survey. Int. J. Mach. Learn. & Cyber. 2(2011) 107-122.

[10] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew. Extreme learning machine: theory and applications, Neurocomputing. 70(2006) 489-501.

[11] G.-B. Huang, H. Zhou, X. Ding and R. Zhang. Extreme Learning Machine for Regression and Multi-Class Classification. IEEE Trans. Syst. Man, Cybern. B, Cybern. 42(2) (2012) 513-529.

[12] S. A. Billings, S. Chen, M. J. Korenberg. Identification of MIMO nonlinear systems using a forward-regression orhtogonal estimator. International Jorunal Control. 49(6) (1989) 2157-2189.

[13] S. Chen, P. M. Grant, C. F. N. Cowan, Orthogonal least-squares algorithm for training multioutput radial basis function networks, IEE Proceedings-F. 139 (6) (1992) 378-384.

[14] K. Li, J. Peng, G. W. Irwin, A fast nonlinear model identification method. IEEE Trans. Autom. Control. 50(8) (2005) 1211-1216.

[15] K. Li, J.Peng and E-W. Bai. A two-stage algorithm for identification of nonlinear dynamic systems. Automatica. 42(7) (2006) 1189-1197.

[16] D. Du, K. Li, M. Fei. A fast multi-output RBF neural network construction method. Neurocomputing. 73 (2010) 2196-2202.

[17] Z. Aydin, A. Singh, J. Bilmes, W. Noble. Learning Sparse models for a dynamic Bayesian network classifier of protein secondary structure. BMC Bioinformatics. 12(154) (2011) 1-21.

[18] S. Chen, X. Hong, C. Harris, P. Sharkey. Sparse modeling using orthogonal forward regression with PRESS statistic and regularization. IEEE Transactions on Systems, Man, And Cybernetics-Part B: Cybernetics. 34(2) (2004) 898-911.

[19] D. J. C. MacKay. Bayesian Interpolation, Neural Comput. 4(3) (1992) 415-447.

[20] M. T. Tipping. Sparse Baesian learning and the relevance vector machine. Journal of Machine Learning Research. 1(2001) 211-244.

[21] Jayadeva, R. Khemchandani and S. Chandra. Regularized least squares support vector regression for the simultaneous learning of a function and its derivatives. Information Science. 178(2008) 3402-3414.

[22] A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. Technometrics. 12(1) (1970) 55-67.

[23] S. Chen. Local regularization assisted orthogonal least squares regression. Neurocomputing. 2006, 69: 559-585.

[24] S. Chen. Multi-output regression using a locally regularised orthogonal least-squares algorithm. IEE Proc. Vis. Image Signal Process. 149(4) (2002) 185-195.

[25] K. Li, J. Peng, and E-W Bai. Two-stage mixed discrete-continuous identification of radial basis function (RBF) neural models for nonlinear systems. IEEE Trans. Circuits Syst. I, Reg. Papers. 56 (3) (2009) 630-643.

[26] S. Chen, C. F. N. Cowan and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Transactions on Neural Networks. 2(2) (1991) 302-309.

[27] C. R.Rao and S. K. Mitra. Generalized inverse of matrices and its applications, Wiley, New York, 1971.

[28] I. J. Leontaritis, and S. A. Billings. Model selection and validation methods for non-linear systems, International Journal of Control. 45 (1987) 311-341.